НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В.Коваль
(підпис) (ініціали, прізвище)
“___”_____2019 р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки
6.050101 “Комп’ютерні науки’’

на тему: Нейронні мережі для синтезу мовлення

Виконав: студент 4 курсу, групи ТР-52

Вишняк Олександр Миколайович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник доцент, к.т.н. Стативка Юрій Іванович _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 6.050101 “Комп’ютерні науки’’

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2019 р.

ЗАВДАННЯ
на дипломну роботу студенту

Вишянк Олександра Миколайовича
(прізвище, ім’я, по батькові)

1. Тема роботи Нейронні мережі для синтезу мовлення
керівник роботи доцент, к.т.н. Стативка Юрій Іванович
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”___”_____ 201__ р. № _____

2. Строк подання студентом роботи ___ _____ 201__ р.
3. Вихідні дані до роботи 32-бітний аудіо файл формату
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) проаналізувати існуючі програмні рішення та можливі засоби реалізації систем перетворення тексту в мовлення обґрунтувати на основі нейронних мереж, проаналізувати які методи генерації тексту більше підходять до даної задачі розробка застосунку для генерації мовлення з вхідного тексту

5. Перелік ілюстраційного матеріалу (з точним зазначенням обов'язкових креслень)
1. Сучасний стан системи TTS. 2. Нейронні мережі так їх типи. 3. Опис нейронних мереж які зарекомендували себе у світі. 4. Досвід навчання системи.
6. Публікації: XVII міжнародна науково-практична конференція молодих вчених та студентів «Сучасні проблеми наукового забезпечення енергетики», м. Київ, КПІ ім.Ігоря Сікорського, 23-26 квітня 2019р.

Дата видачі завдання "___"_____ 201__ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Вивчення та аналіз задачі		
2.	Розробка архітектури та загальної структури системи		
3.	Розробка структур окремих підсистем		
4.	Підготовка матеріалів		
5.	Програмна реалізація системи		
6.	Захист програмного продукту		
7.	Оформлення пояснювальної записки		
8.	Передзахист		
9.	Захист		

Студент _____ Вишняк О. М.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Стативка Ю.І..
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Мета роботи — система генерації мовлення із заданого тексту, з використанням методів які базуються на нейронних мережах. Було використано модель DCTTS, на основі якої був створений веб-застосунок, за допомогою якого користувача може перетворити текст в мовлення. При чому користувач може як прослухати так і завантажити даний аудіо файл. Генероване мовлення чітке і розбірливе, проте все ж таки має недоліки

Записка містить 66 сторінок, 14 рисунків, 3 таблиці, 4 додатків і 18 посилань.

Ключові слова: TTS, DCTTS , WaveNet, DeepVoice,T acotron, RNN, CNN, сервер для генерації тексту.

,

ABSTRACT

The purpose of the work is a system for generating speech from a given text, using methods based on neural networks. The DCTTS model, based on which a web application was created, allows the user to convert text to speech. At that, the user can both listen and download this audio file. The generated broadcast is clear and legible, but it still has disadvantages.

The note contains 66 pages, 14 figures, 3 tables, 4 attachments and 18 links.

KEYWOR: TTS, DSTTS, DCTTS , WaveNet, DeepVoice,T acotron, RNN, CNN, server for text to speech

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

TTS — Text to Speech

DCTTS —Deep Convolutional Text to Speech

LSTM — Long Short-Term Memory

RNN — Recurrent Neural Network

SSRN — Spectrogram Super-resolution Network

CNN — Convolutional Neural Network

ВСТУП

В наш час, коли технології розвиваються надзвичайно швидко, надзвичайно швидко й збільшується кількість інформації, проте ця інформація, зазвичай, зберігається саме у вигляді тексту. Все ж, на даний час, все більше набирають популярність саме аудіо та відео формат, адже це більш практичній спосіб засвоєння інформації, бо не завжди є можливість дістати книгу, а от дістати навушники і слухати аудіо-книги, новини можна і під час ранкової пробіжки, в громадському транспорті, прогулянці та в будь-якому завдані, яке не вимагає сильної концентрації. Це економить час, що є найважливішою перевагою цього формату.

Популярність аудіо формату спричинила необхідність його використання, тобто задача звелася, до того, що потрібно було генерувати мовлення, в великих обсягах. Синтез мови на сьогоднішній день застосовується в багатьох областях. Це і голосові асистенти, і IVR-системи (системи голосових меню), і розумні будинки, і т.і.

Досягнення істотних результатів у синтезі мови стало можливим лише зі зростанням потужності обчислювальної техніки, а також з розвитком математичних методів та програмних продуктів запису, дослідження та обробки цифрової звукової інформації. В основу досягнень цих результатів лягли дослідження, що проводилися вченими протягом 60-х – 90-х років XX століття [1].

На сьогоднішній день найбільш розповсюдженим та одним із найбільш ефективних підходів до створення систем комп’ютерного озвучування природних мов є Text-to-Speech синтез. Про це свідчать передові досягнення світових досліджень, що проводяться у даній галузі лабораторіями провідних компаній, які займаються розробкою програмного забезпечення (наприклад, AT&T Labs’ Natural Voices™ компанії AT&T Labs [2]; Apple’s Speech Recognition and Speech Synthesis Technologies [3]; IBM Text-to-Speech Research компанії IBM [4]; Microsoft Reader компанії Microsoft [5]).

У середині 80-х р. була запропонована концепція TTS синтезу. TTS синтез – це комп’ютерна система, яка довільну текстову інформацію перетворює в еквівалентну звукову мовну інформацію, синтезуючи нові слова, словосполучення, речення тощо [6].

Технологія TTS синтезу дозволяє комп’ютерам перетворювати довільний текст у чутну мову з метою можливості доставити текстову інформацію за допомогою голосових повідомлень. Ключова ціль TTS застосувань у системах зв’язку складається із представлення голосом текстових повідомлень [7].

В даній роботі я спробою пояснити, як останні досягнення в синтезі мовлення, використовуючи методи глибокого навчання, застосовуються для створення більш природнього звучання мови.

Звіт з переддипломної практики містить 5 розділів.

У першому розділі описується постановка задачі ефективного синтезу мовлення.

У другому розділі вказується сучасний стан, даної проблеми.

У третьому розділі описуються архітектури нейронних мереж які зазвичай використовуються в таких випадках.

У четвертому розділі описуються нейронні мережі, які зарекомендували себе у світі.

У п’ятому розділі дано опис даних про навчання системи.

У шостому розділі описані основні засоби розробки.

У сьомому розділі описується досвід навчання та робота з методами синтезу мовлення

1 ПОСТАНОВКА ЗАДАЧІ ЕФЕКТИВНОГО СИНТЕЗУ МОВЛЕННЯ

З ростом популярності інтернет-ЗМІ, блогів і тематичних співтовариств обсяг генерованого контенту збільшується в геометричній прогресії. Вже зараз багато користувачів фізично не встигають обробляти свою підписку, раз у раз натискаючи на “червону кнопку”, що видаляє всі непрочитані статті віком понад 48 годин. Зізнатися, для мене, як дуже активного – як в онлайні, так і в офлайні людини, повністю прочитані канали - в буквальному сенсі недозволена розкіш.

Через мобільні пристрої ми можемо читати статті, перебуваючи далеко від стаціонарного комп'ютера або ноутбука. Але складніше, якщо ви сидите за кермом автомобіля, займаєтеся спортом або просто перебуваєте в пішій прогулянці. З екрану в ці моменти читати текст незручно, хоча цей час можна було б також використовувати для отримання інформації.[8]

Цілком логічним, на мій погляд, тут є рішення отримувати матеріали своєї підписки в форматі аудіо: перетворювати текст в мову, завантажувати файли в плеєр або мобільний телефон і слухати статті майже в будь-який час і в будь-якому місці.

Саме по собі завдання, на мою думку, дуже зрозуміле: написаний текст повинен вимовлятися так, як це б зробила людина. Найчастіше нам потрібно відтворити аудіо-інформацію, яка не була записана заздалегідь.Для цього найкраще використовувати технологію TTS (“перетворення тексту в мову”).

Системи “текст-мова” (TTS, Text-to-Speech), які синтезують текст в мову, існують досить давно: ще в 1961 році на комп'ютері IBM 704 вперше була “штучним” чином відтворена людська мова. З тих часів, звичайно, змінилося багато чого: комп'ютери навчилися синтезувати голос більш реалістично, а на технології TTS було створено безліч програмних продуктів. [9].

Оскільки система динамічно створює необхідні аудіофайли. Для генерації аудіофайлів використовується певний алгоритм, після чого створюється локальний WAV файл.

Розроблена система повинна задовольняти таким критеріям:

– текст повинен вимовлятися так, як це б зробила людина;
– мати оптимальну реалізацію із точки зору впливу на продуктивність;
– синтезування мовлення повинно забирати не надто багато часу.

Для вирішення поставленої задачі будемо використовувати сучасні алгоритми, які і забезпечать високу ефективність нашої системи, тобто згенерований звук, буде максимально схожий, на той, який би записала людина.

2 СУЧАСНИЙ СТАН СИСТЕМ TTS

Декілька років тому, в область синтезу мови, як і в багато інших областей, прийшло машинне навчання. З'ясувалося, що цілий ряд компонентів всієї системи можна замінити на нейронні мережі, що дозволяє не просто наблизитися за якістю до існуючих алгоритмів, а навіть значно їх перевершити [10].

Щоб зрозуміти, чому сьогодні використовуються методи глибокого навчання для формування мовлення, важливо зрозуміти, як традиційно робиться генерація мовлення. Існує два специфічних способи перетворення тексту в мову (TTS) – Параметричні TTS і конкатенативні TTS.

Важливо також визначити два терміни, про які йдеться в char2wav, щоб судити про якість згенерованої мови. Розбірливість і природність. Розбірливість – це якість звуку, що генерується. Чи це чисто, чи можна слухати? А природність – це якість мови, що генерується. Чи звучить він емоційно? Чи має мова правильний темп і вимову?

2.1 Конкатенативні TTS

Цей метод заснований на попереднім записом коротких аудіо-фрагментів, які потім об'єднуються для створення зв'язного мовлення. Вона виходить дуже чиста і ясна, але абсолютно позбавлена емоційної і інтонаційної складових, тобто звучить неприродньо. А все тому, що неможливо отримати аудіо запис всіх можливих слів, вимовлених у всіх можливих поєднаннях емоцій і просодії.

Системи конкатенативного TTS синтезу оперують мінімальними мовними даними – конкатенаційними елементами синтезу. Тут принциповим є вибір сегментів синтезу, від яких залежатиме природність звучання, уривчастість та розбірливість синтезованої мови.

Спочатку обробляється вхідна Текстова Інформація (ТІ). Виділяються ознаки та параметри конкатенативного синтезу, проводиться аналіз та сегментація тексту на текстові елементи.

Операції, що містяться в модулі цифрової обробки сигналу, є комп'ютерним аналогом динамічного контролю артикуляторних м'язів та вібруючої частоти голосових зв'язок таким чином, що вихідний сигнал підбирає вхідні умови. Для того щоб зробити це правильно, модуль обробки цифрового сигналу повинен деяким чином враховувати обмеження, оскільки для розуміння фонетичні переходи важливіші, ніж сталі стани [6]. Необхідні звукові елементи (еквівалентні текстовим елементам) синтезу мови (завчасно записані мовні сигнали) піддаються обробці спеціальними методами обробки звукових сигналів. Оброблені сигнали надходять на блок озвучення, де і відбувається генерація вихідного звукового сигналу з подальшим його озвученням та записом у звукові файли.

Структура модуля конкатенації визначається розміром та розмірністю бази даних природних звукових мовних елементів, оскільки безпосередньо залежить від природних даних, тим самим забезпечуючи високу природність звучання синтезованої мови. Тому при підвищенні природності звучання синтезованої мови буде зростати і розмірність елементної бази синтезу [8]. Схематично робота конкатенативних систем TTS синтезу представлена на рис. 2.1.

Акустичний процесор генерує та озвучує акустичний файл, відповідний вхідному тексту. Генерація мовного файлу полягає в конкатенації вибраних оптимальних елементів мовної БД [11].

Системи конкатенативного синтезу маніпулюють мовними сигналами як сукупностями мовних елементів. Тому для підвищення рівня природності звучання згенерованих мовних сигналів необхідно розробити таку структуру сегментації/конкатенації природних мовних елементів, яка б враховувала та використовувала ознаки природності звучання мовних сигналів [12].

Конкатенативні системи вимагають величезних баз даних і жорсткого кодування комбінацій для формування слів. Розробка надійної системи займає дуже багато часу [9].

2.2 Параметричні TTS

Застосування конкатенативного TTS обмежено через високі вимоги до даних і часу розробки. Тому був розроблений статистичний метод, який досліджує саму природу даних. Він генерує мову за допомогою комбінування таких параметрів, як частота, спектр амплітуд і т.і.

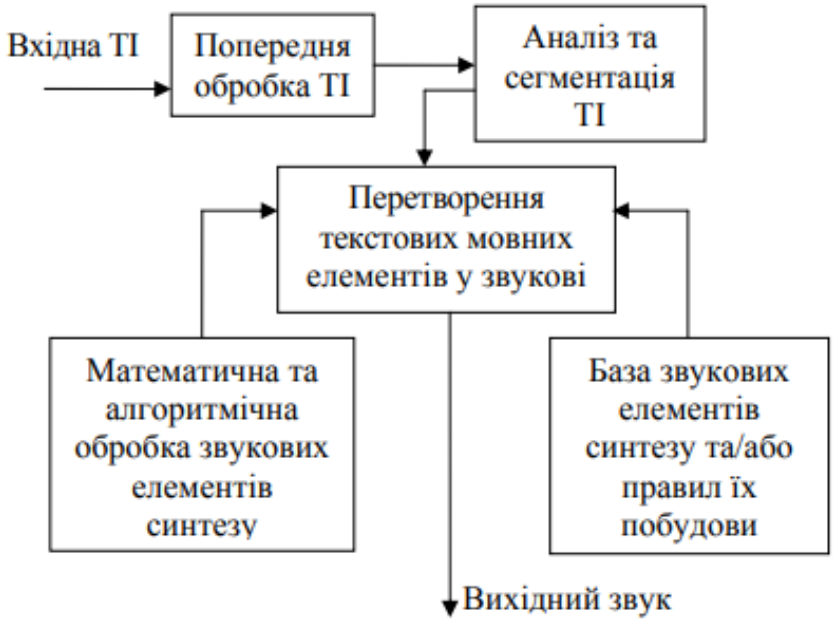

Рисунок 2.1 – Етапи роботи конкатенативної системи озвучення текстової інформації згідно з концепцією TTS синтезу

Параметричний синтез складається з двох етапів :

– спочатку з тексту витягуються лінгвістичні ознаки, наприклад, фонеми, тривалість і т.д;

– потім для вокодера (системи, що генерує wave-форми) витягуються ознаки, які представляють відповідний мовний сигнал: кепстральні коефіцієнти частоти та лінійну спектрограму.

Ці, налаштовані вручну, параметри поряд з лінгвістичними особливостями передаються в модель вокодера, а той виконує безліч складних перетворень для

генерування звукової хвилі. При цьому вокодер оцінює параметри мови, такі як фаза, просодія, інтонація та інші.

Якщо ми можемо апроксимувати параметри, які визначають мову на кожній її одиниці, тоді ми зможемо створити параметричну модель. Параметричний синтез вимагає значно менше даних і важкої роботи, ніж конкатенативні системи.

Теоретично все просто, але на практиці виникає багато артефактів, що призводять до приглушеної мови з “металічним” призвуком, що зовсім не схоже на природне звучання.

Справа в тому, що на кожному етапі синтезу ми чітко задаємо деякі ознаки і сподіваємося отримати реалістичне мовлення. Але вибрані дані базуються на нашому розумінні мови, але ж людське знання не абсолютне, тому взяті ознаки не обов’язково будуть найкращим можливим рішенням.

І тут на сцену виходять нейронні мережі у всій своїй красі.

2.3 Синтез мовлення з використанням нейронних мереж

Глибокі нейронні мережі є потужним інструментом, який, теоретично, може апроксимувати функцію довільної складності, тобто, відобразити деякий простір вхідних даних X в простір вихідних даних Y. У контексті нашої задачі це будуть, відповідно, текст і аудіо з промовою.

Тепер, працюючи над цим припущенням, система природного звучання Text-to-Speech повинна мати вхід X як рядок тексту, а Y – звуковий сигнал.

Вона не повинна використовувати будь-які ручні конструктивні особливості, а навпаки вивчати нові високо вимірні можливості для представлення того, що робить мову, схожою на людську.

Таким чином при використанні нейронних мереж в системах TTS, ми отримаємо систему яка автоматично калібрується на основі передбачень, які генерується нейронною мережею.

2.4 Висновки до розділу 2

У розділі було розглянуто два основних типи систем TTS, були розглянуті їхні недоліки та переваги. Також було розглянуто чому в один із типів систем інтегрувалися нейронні мережі.

Внаслідок цього був вибраний параметричний тип TTS, який буде оснований на нейронній мережі , для досягнення кращої якості генерованого звуку.

3 НЕЙРОННІ МЕРЕЖІ ТА ЇХ ТИПИ

Штучна нейронна мережа є концептуальною моделлю біологічної нейронної мережі і складається з пов'язаних між собою різним чином шарів штучних нейронів, які організовують загальну активну структуру і функціонально впливають на роботу один одного. У більшості архітектур штучних нейромереж активність нейрона визначається перетворенням зовнішнього сумарного впливу інших нейронів на даний нейрон.

З моменту свого зародження технології штучних нейронних мереж розвивалися досить відокремлено від класичних методів, нерідко докорінно змінюючи уявлення про предмет і проблематику теорії машинного навчання і розпізнавання об’єктів, залишаючи значний вплив на теоретичний, термінологічний і методологічний апарати цих дисциплін. Через деякий час після розвитку базових моделей штучних нейронних мереж, відбувся значний поділ науки про нейромережі на види топологій архітектури мереж і методи навчання мереж.

У більшості архітектур штучних нейронних мереж функції активації нейронів фіксовані, а ваги синапсів є параметрами мережі. Деякі входи нейронів є зовнішніми входами сукупної мережі, а деякі виходи нейронів - виходами сукупної мережі.

За всіма видами архітектур нейронних мереж, які виникають останнім часом, встежити непросто. Тому що архітектура, зазвичай розробляється саме для специфічних проблем , які потребують .нестандартних рішень. Розглянемо декілька типів нейронних мереж, які найбільше підходять для нашого системи TTS.

3.1 Згорткові нейронні мережі (CNN)

Кардинально відрізняються від інших мереж (рисунок 3.1). Вони використовуються в основному для обробки зображень, іноді для аудіо та інших

видів вхідних даних. Типовим способом застосування CNN є класифікація зображень: якщо на вхід подається зображення кішки, мережа видасть «кішка», якщо картинка собаки - «собака». Такі мережі зазвичай використовують «сканер», що не обробляє всі дані за один раз. Наприклад, якщо у вас є зображення 200x200, ви захочете будувати шар мережі з 40 тисяч вузлів. Замість цього мережа вважає квадрат розміру 20x20 (зазвичай з лівого верхнього кута), потім зрушиться на 1 піксель і вважає новий квадрат, і т.д.

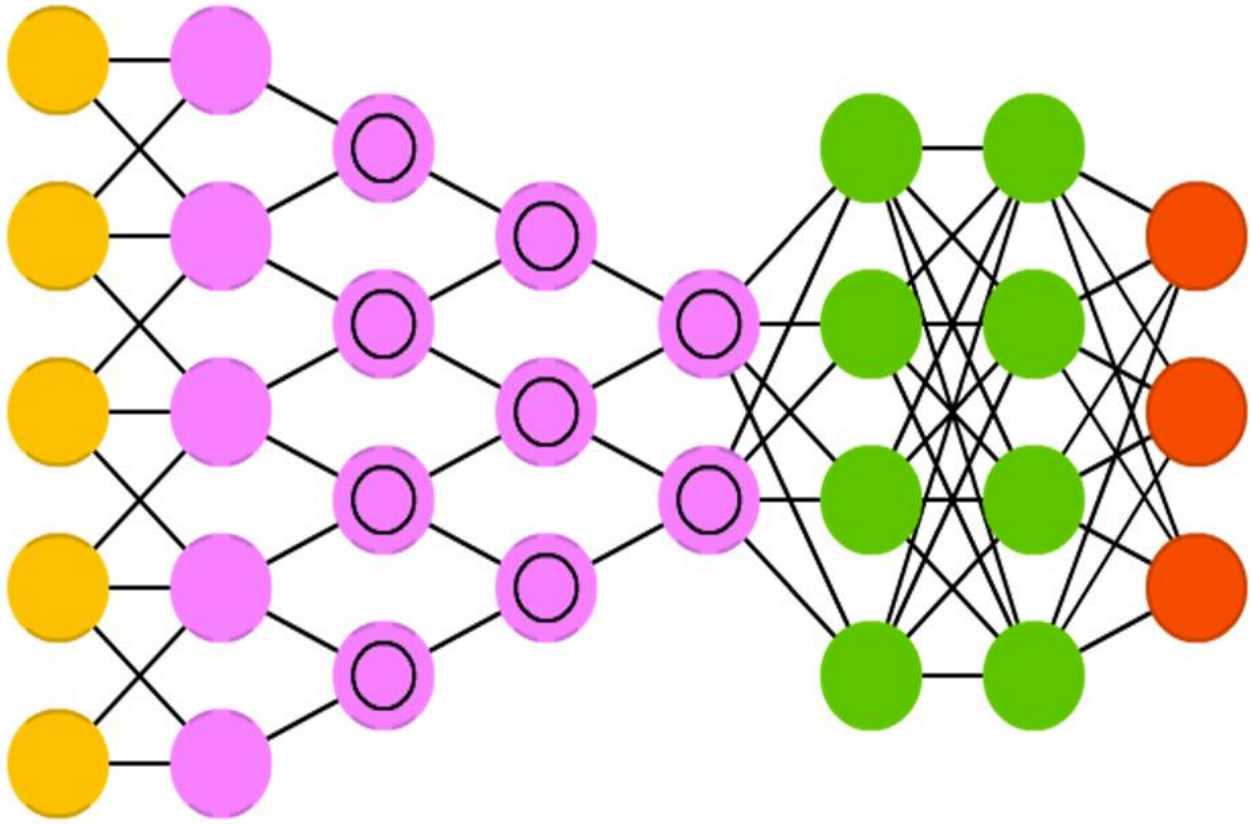

Рисунок 3.1 – Схематичне зображення згорткових нейронних мереж

Зауважте, що ми не розбиваємо зображення на квадрати, а скоріше повземо по ньому. Ці вхідні дані потім передаються через згортаючі шари, в яких не всі вузли з'єднані між собою . Замість цього кожен вузол з'єднаний тільки зі своїми найближчими сусідами. Ці шари мають властивість стискуватися з глибиною, причому зазвичай вони зменшуються на який-небудь з дільників кількості вхідних даних (наприклад, 20 вузлів в наступному шарі перетворяться в 10, в наступному – в 5), часто використовуються ступеня двійки.

На рисунках 3.2 та 3.3 зображено операція згортки при застосуванні фільтру kernel для зображення .

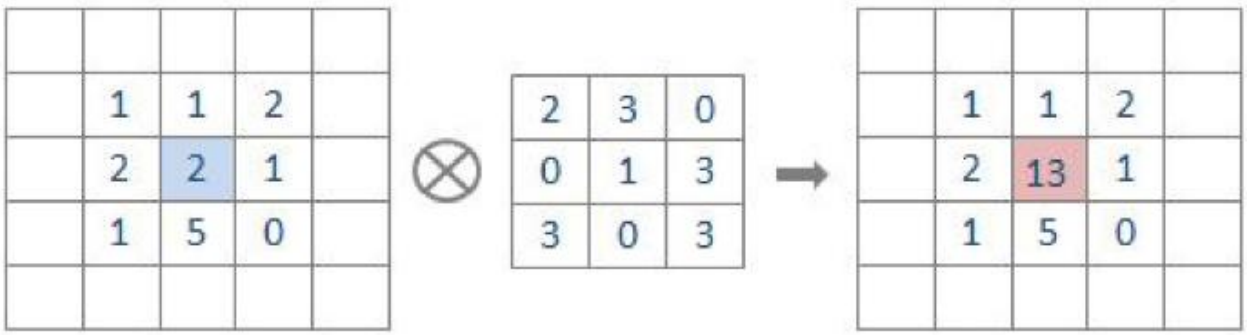

Рисунок 3.2 – Візуалізація операції згортки

Крім згортальних шарів є також так звані шари об'єднання (pooling layers). Об'єднання - це спосіб зменшити розмірність одержуваних даних, наприклад, з квадрата 2x2 вибирається і передається найбільш червоний піксель. На практиці до кінця CNN прикріплюють FFNN для подальшої обробки даних.

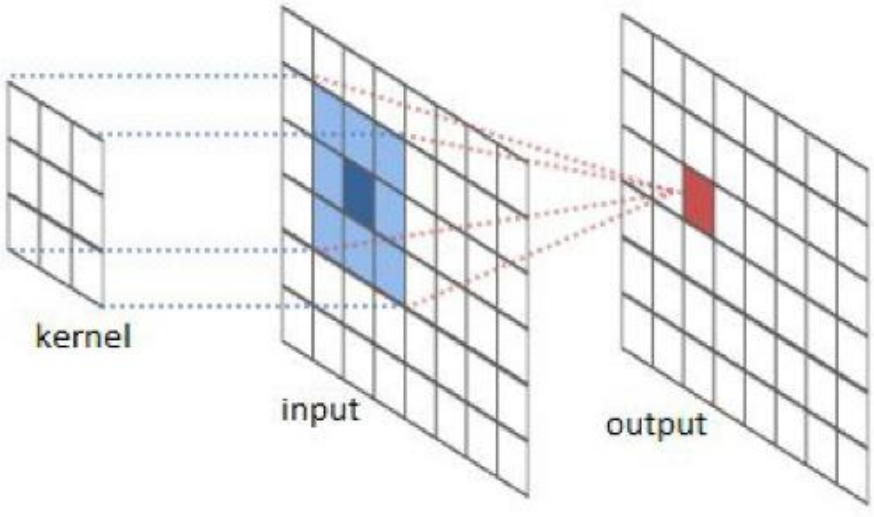

Рисунок 3.3 – Графічне зображення операції згортки

Є матриця, яка називається ядром фільтра (на малюнку kernel). Для Blur це будуть всі одиниці. Є зображення. Ця матриця накладається на шматочок

зображення, відповідні елементи просто перемножуються, результати складаються і записуються в центральну точку.

3.2 Розгортаючі нейронні мережі (DNN)

Так звані розгортаючі нейронні мережами – це згорткові нейронні мережі навпаки (рисунок 3.4). Уявіть, що ви передаєте мережі слово "кішка" і навчаєте її генерувати картинки кішок шляхом порівняння одержуваних картинок з реальними зображеннями кішок. DNN теж можна об'єднувати з FFNN.

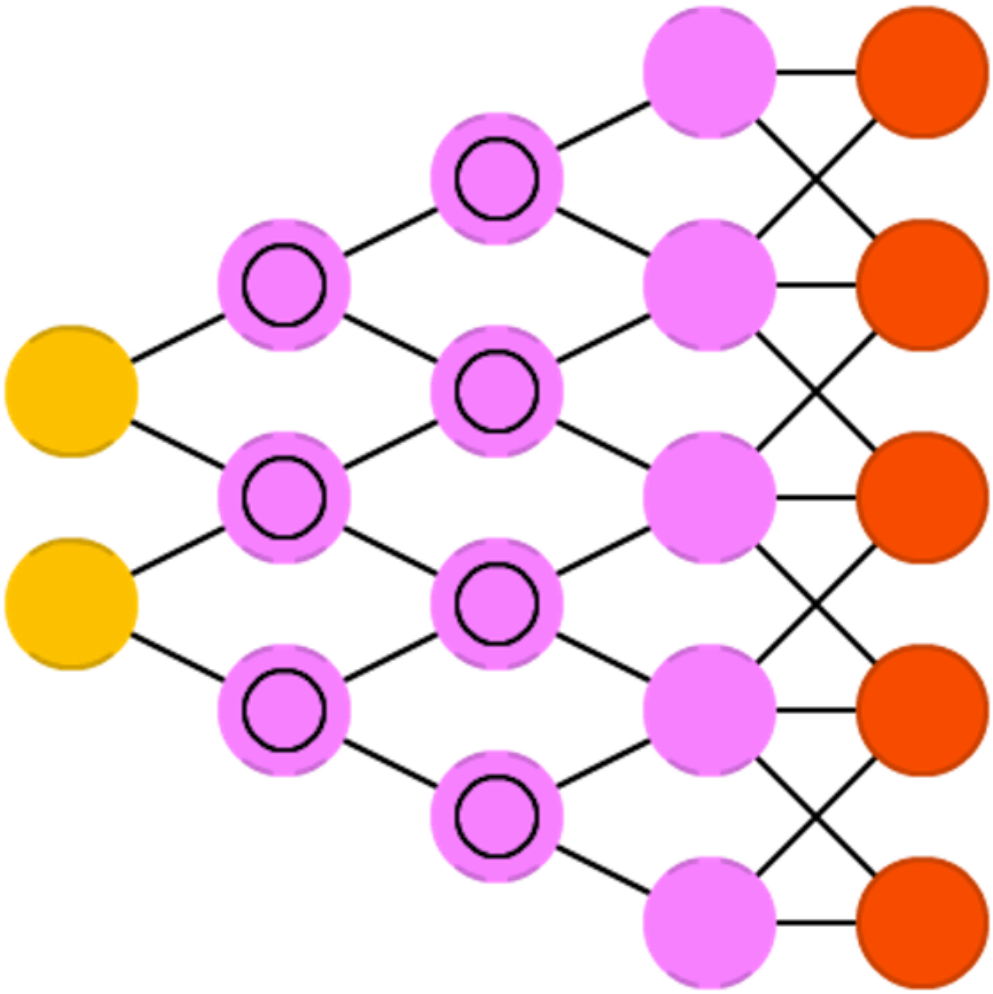

Рисунок 3.4 – Схематичне зображення розгортаючих нейронних мереж

Варто зауважити, що в більшості випадків мережі передають не рядок, а бінарний класифікацію вектор: наприклад, <0, 1> - це кішка, <1, 0> - собака, а <1,

1> - і кішка, і собака. Замість шарів об'єднання, які часто зустрічаються в CNN, тут присутні аналогічні зворотні операції, зазвичай інтерполяцію або екстраполяцію.

3.3 Рекурентні нейронні мережі (RNN)

Це ті ж мережі прямого поширення, але зі зміщенням в часі: нейрони отримують інформацію не тільки від попереднього шару, але і від самих себе в результаті попереднього проходу. Отже, тут важливий порядок, в якому ми подаємо інформацію і навчаємо мережу: ми отримаємо різні результати, якщо спочатку згодуємо їй "молоко", а потім "банка", або якщо спочатку "банка", а потім вже "молоко".

У RNN є одна велика проблема – це проблема зникання (або вибухового) градієнта: в залежності від використовуваної функції активації інформація з часом втрачається, так само як і в дуже глибоких мережах прямого поширення. Здавалося б, це не така вже й серйозна проблема, так як це стосується тільки ваг, а не станів нейронів, але саме в вагах зберігається інформація про минуле; якщо вага досягне значення 0 або 1 000 000, то інформація про минулі стани стане не дуже інформативною. RNN можуть використовуватися в найрізноманітніших областях, так як навіть дані, не пов'язані з плином часу (не звук або відео) можуть бути представлені у вигляді послідовності. Картинка або рядок тексту можуть подаватися на вхід по одному пікселю або символу, так що вага буде використовуватися для попереднього елемента послідовності, а не для того, що трапилося X секунд тому. У загальному випадку, рекурентні мережі хороші для продовження або доповнення інформації, наприклад, автодоповнення.

4 ОПИС НЕЙРОМЕРЕЖ ЯКІ ЗАРЕКОМЕНДУВАЛИ СЕБЕ У СВІТІ

4.1 Метод синтезу голосового мовлення WaveNets

Зазвичай дослідники уникають моделювання аудіо семплів, тому що їх потрібно генерувати дуже багато: до 16000 семплів в секунду або більше, строго певної форми в будь-яких тимчасових масштабах (рисунок 4.1).

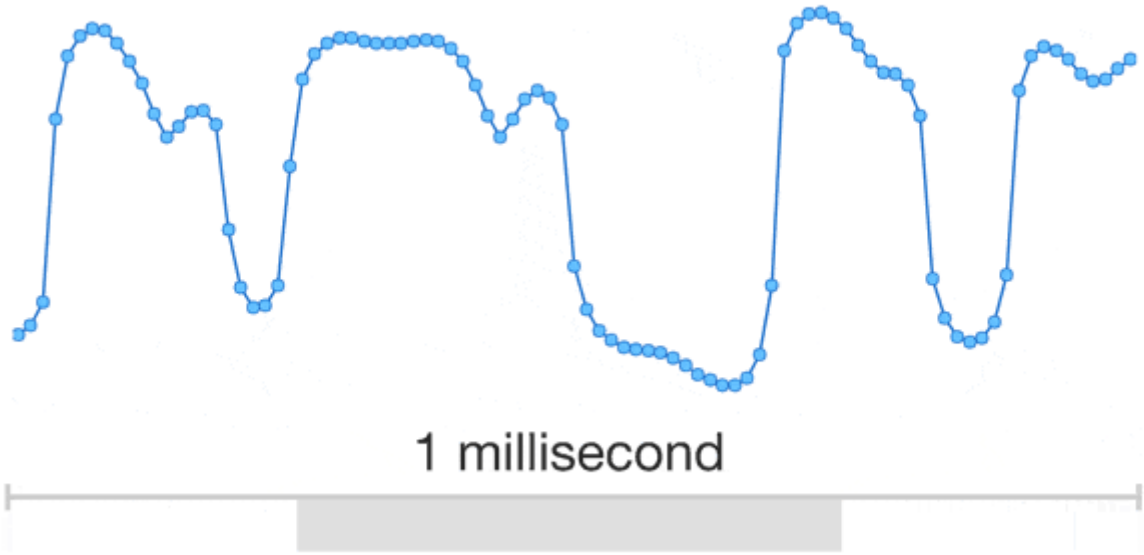

Рисунок 4.1 – Приклад генерації семплів на одну 1мс.

Побудова авторегресійної моделі, в якій кожен семпл залежить від усіх попередніх – це непросте завдання. Проте, наші моделі PixяelRNN і PixelCNN, опубліковані раніше в цьому році, показали, що можливо генерувати складні природні зображення не тільки по одному пікселю за момент часу, але і по одному колірному каналу за момент часу, що вимагає тисячі пророкувань на зображення (рисунок 4.2). Це надихнуло нас адаптувати двомірні PixelNets в одновимірну WaveNet [2].

Кожен шар в цій мережі має свій фактор розширення, завдяки якому її рецептивне поле, тобто частина інформації, яку обробляють нейрони, зростає експоненційно . По суті, це дозволяє програмі охоплювати відразу велику кількість часових кроків. У нейромережі також передбачений зворотний зв'язок, тому кожен наступний звук машинної мови генерується на основі безлічі попередніх.

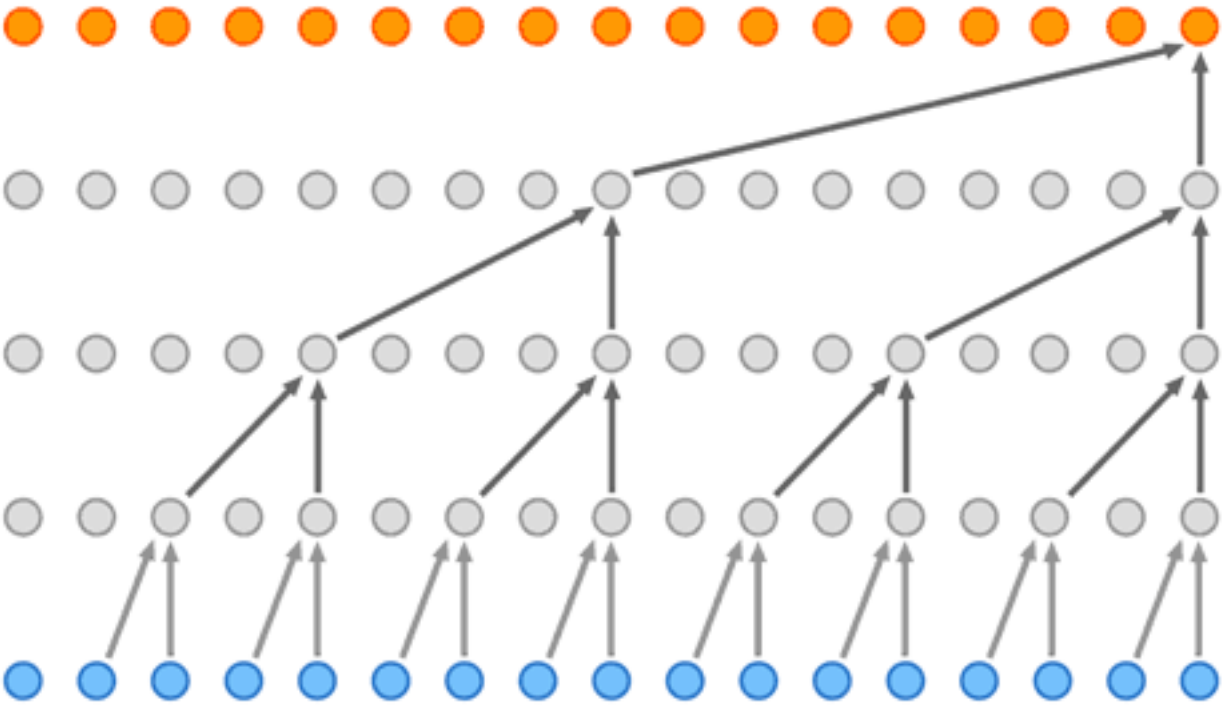

Рисунок 4.2 – Приклад генерації семпла, на основі попередніх пророкувань нейромережі

Під час навчання вхідними послідовностями є звукові хвилі від прикладів записи голосу. Після тренування можна за допомогою мережі генерувати синтетичні фрази. На кожному кроці семплинга значення обчислюється з імовірнісного розподілу розрахунку мережі. Потім це значення повертається на вхід і робиться нове передбачення для наступного кроку. Створення семплів таким чином є досить ресурсномістким завданням, але ми з'ясували, що це необхідно для генерації складних, реалістичних звуків.

4.2 Метод синтезу голосового мовлення Tacotron 2

Tacotron 2 являє собою не одну мережу, а дві: Feature prediction net і NN-vocoder WaveNet.

Tacotron2 – це архітектура “послідовность до послідовності”. Вона складається з кодувальника (encoder), що створює деякий внутрішнє уявлення про вхідний сигнал (символьних токенах), і декодувальник (decoder), який перетворює це уявлення в мел-спектрограму (рисунок 4.3). Також вкрай важливим елементом мережі є так званий PostNet, покликаний поліпшити спектрограму, яка була згенерована декодером.

Першим шаром кодувальника є Embedding-шар. Він на підставі послідовності натуральних чисел, що представляють символи, створює багатовимірні (512-мірні) вектори.

Далі embedding-вектори подаються в блок з трьох одновимірних згортаючих шарів. Кожен шар включає в себе 512 фільтрів довжиною 5. Це значення є хорошим розміром фільтра в даному контексті, тому що захоплює певний символ, а також два попередніх і два наступні його сусіда. За кожним згортаючим шаром слідує нормалізація по міні-батчу і ReLU-активацій.

Тензори, отримані після згортаючого блоку, подаються на двонаправлені LSTM-шари, по 256 нейронів у кожному. Результати прямого і зворотного проходу конкатенуються.

Декодер має рекурентну архітектуру, тобто на кожному наступному кроці використовуються вихідні дані з попереднього кроку. Тут ними буде один фрейм спектрограми. Ще одним важливим, якщо не ключовим, елементом даної системи є механізм м’якого (навчаної) “уваги” – відносно новий і набирає все більшої популярності прийом. На кожному кроці декодера увагу для формування контекстного вектора і поновлення ваги уваги використовує:

– проекцію попереднього прихованого стан RNN-мережі декодера наповнозв'язну шар;

– проекцію вихідних даних кодувальника на повнозв’язний шар,а також адитивні (що накопичуються на кожному часовому кроці роботи декодера) ваги уваги.

Ідею “уваги” слід розуміти так: “частину даних кодувальника які слід використовувати на поточному кроці декодера” (рисунок 3.4).

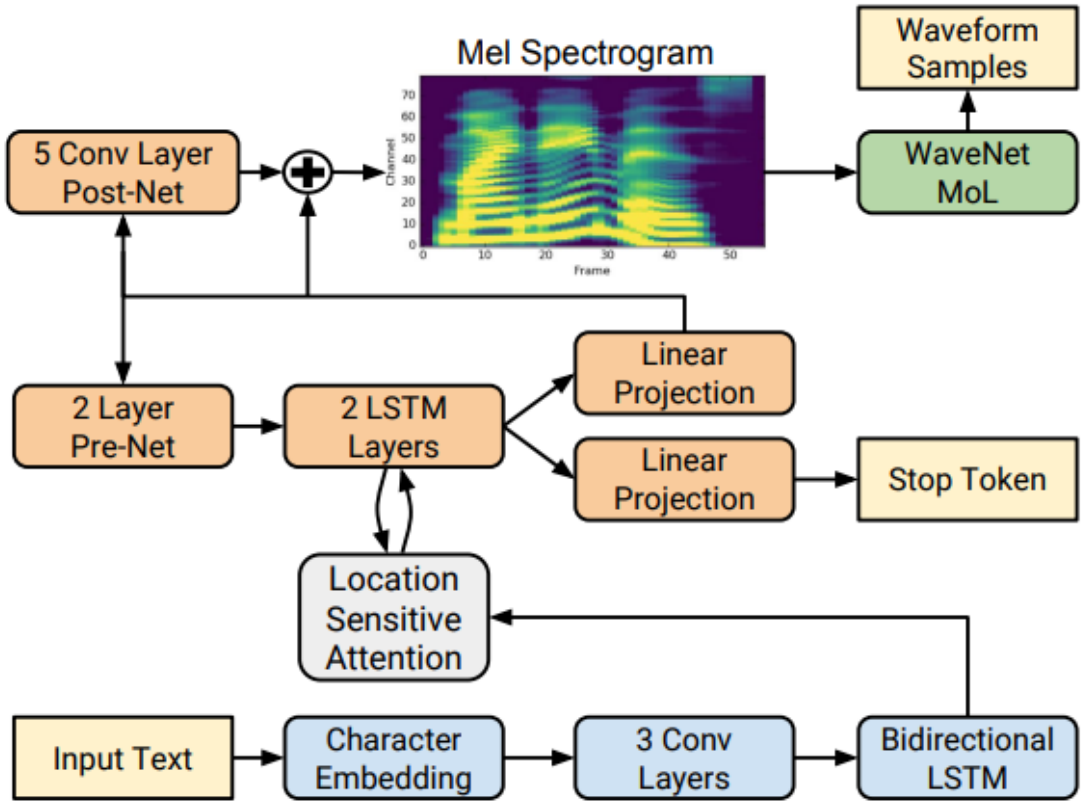

Рисунок 4.3 – Архітектура нейронної мережі Tacotron 2

Тепер розглянемо алгоритм роботи.

Спочатку вихідні дані декодера з попереднього тимчасового кроку подаються в невеликий модуль PreNet, що представляє собою стек з двох повнозв’язних шарів по 256 нейронів кожен, чергуються з dropout-шарами з рейтом 0,5. Відмінною рисою цього модуля є те, що dropout використовується в ньому не тільки на етапі навчання моделі, але і на етапі виведення

Вихідні дані PreNet в конкатенації з контекстним вектором, отриманим в результаті роботи механізму уваги, подаються на вхід в односпрямовану двошарову LSTM-мережу, по 1024 нейрона в кожному шарі.

Потім конкатенація вихідних даних LSTM-шарів з тим же (а можливо, і іншим) контекстним вектором подається в повнозв’язний шар з 80 нейронами, що

відповідає кількості каналів спектрограми. Цей фінальний шар декодера і формує передбачену спектрограму фрейм за фреймом. А вже його вихідні дані подаються в якості вхідних на наступний часовий крок декодера в PreNet.

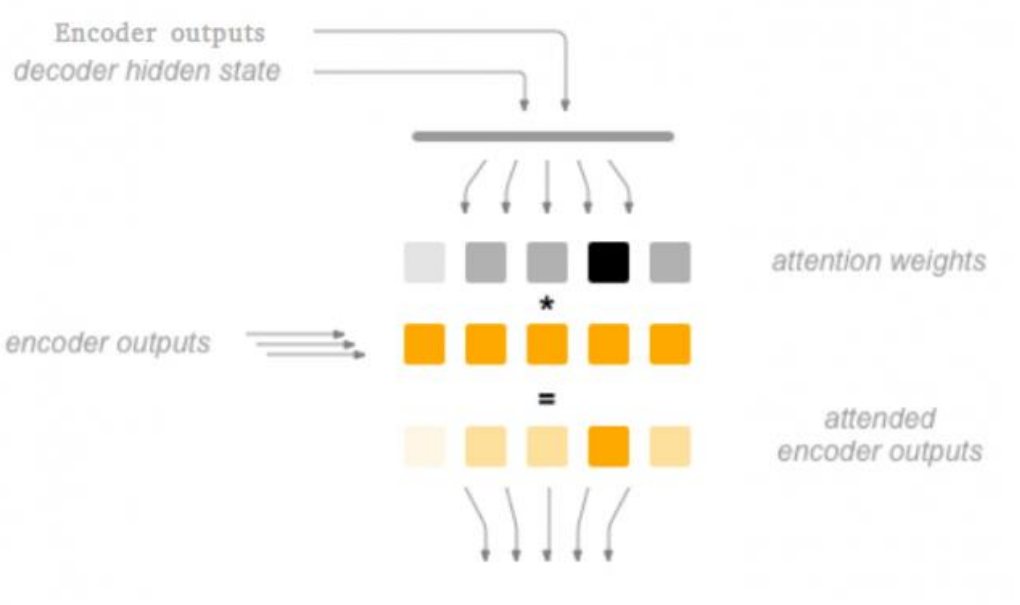

Рисунок 4.4 – Схема роботи механізму “уваги”

Крім проекції на 80-нейронний повнозв’язний шар конкатенація вихідних даних LSTM-шарів з контекстним вектором подається в повнозв’язний шар з одним нейроном, після якого слід сигмоїдна активація - це шар “stop token prediction”. Він пророкує ймовірність того, що фрейм, створений на даному етапі декодера є завершальним. Цей шар призначений для того, щоб на етапі виведення моделі генерувати спектрограму не фіксованої, а довільної довжини. Тобто на етапі виведення цей елемент визначає кількість кроків декодера. Його можна розглядати як бінарний класифікатор.

Вихідними даними декодера з усіх його кроків буде з’являться передбачена спектрограма. Однак і це ще не все. Для поліпшення якості спектрограми вона пропускається через модуль PostNet, що представляє собою стек з п'яти одновимірних згортуючих шарів з 512 фільтрами в кожному і з розміром фільтра 5. За кожним шаром (крім останнього) слідує батч-нормалізація і тангенс-активація.

Можемо повернутися до розмірності спектрограми якщо пропустимо вихідні дані post-net через повнозв’язний шар з 80 нейронами і складаємо отримані дані з

початковим результатом роботи декодера. Отримуємо згенеровану з тексту mel-спектрограму.

Всі згортуючи модулі регулюються dropout-шарами з рейт 0,5, а рекурентні шари - більш новим методом zoneout з рейт 0,1. Він досить простий: замість того, щоб подавати на наступний часовий крок LSTM-мережі прихований стан і стан осередку, отримані на поточному кроці, ми замінюємо частину даних значеннями з попереднього кроку. Це робиться як на етапі навчання, так і на етапі виведення. При цьому zoneout-метод піддається тільки прихований шар (що передається на наступний крок LSTM) на кожному кроці, в той час як висновок LSTM-осередка на поточному кроці залишається незмінним.

4.3 Метод синтезу голосового мовлення DCTTS

Наступний метод, який буде досліджуватися називається Deep Convolutional TTS (DCTTS).

На малюнку 4.7 показана загальна архітектура даного методу
Модель Deep Convolutional TTS (DCTTS) включає в себе дві нейронні мережі:

Text2Mel, яка синтезує спектрограму мелодії з вхідного тексту. Це генерація відбувається за допомогою 64 "механізму уваги" ("Attention"), який використовує два енкодера ("TextEnc", "AudioEnc") і декодер ("AudioDec");
Spectrogram Super-resolution Network (SSRN), яка перетворює мел-спектрограму спектрометра в амплітудну спектрограму STFT, при цьому беручи до уваги пропуски кадрів і відновлюючи частоту дискретизації .

Розглянемо нейронну мережу Text2Mel. Мережа TextEnc спочатку кодує вхідний пропозицію $L = [l_1,...,l_n] \in Char^N$ що складається із N символів, в двох матрицях $K,V \in R^{d \times T}$. З іншого боку , мережа AudioEnc кодує великомасштабну спектрограму $S(S_{1:F,1:T}) \in R^{F \times T}$ раніше записаної мови, довжина якого дорівнює T, в матриці $Q \in R^{d \times T}$.

Матриця уваги $N \times T$ визначається таким чином, оцінює, наскільки сильно пов'язані n-й символ l_n і t-й часовий відрізок $S_{1:F,1:T}$

$$A = soft\max_{n-axis}(\frac{K^T Q}{\sqrt{d}}) \tag{4.1}$$

$A_{nt} \sim 1$ має на увазі, що модуль дивиться на n-й символ l_n на часовий інтервал t, і він буде дивитися на l_n або l_{n+1} якщо символи навколо них на наступному часовому інтервалі t+1.

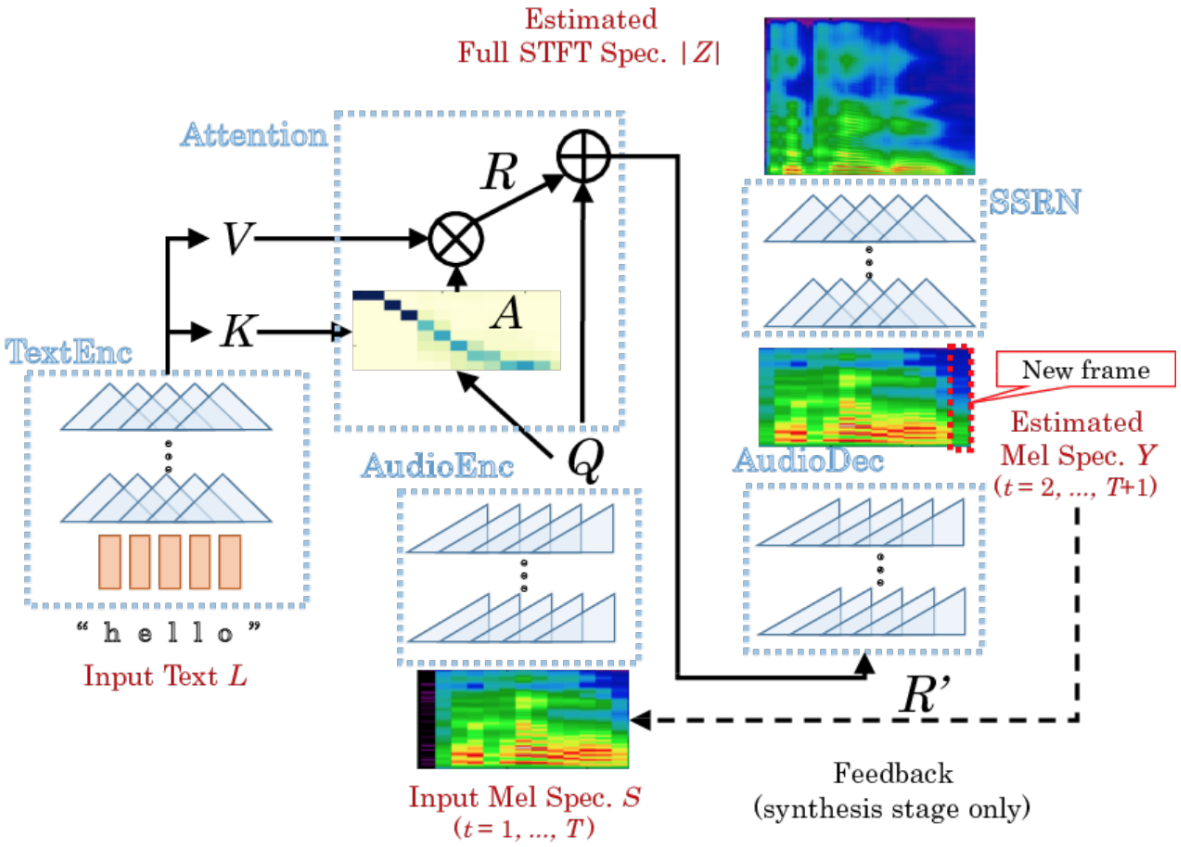

Рисунок 4.5 – Загальна архітектура DCTTS

Щоб цього не сталося, необхідно очікувати, що вони будуть закодовані в n-м V. Таким чином початкове числи $R \in R^{d \times T}$,декодування в наступні кадри $S_{1:F,1:T+1}$, отримуємо так:

$$R \in Att(Q,K,V) := VA \tag{4.2}$$

Результат R об'єднується з кодованим звуком Q,як R=[R,Q]. Потім конкатенування матриця $R' \in R^{2d \times T}$ декодується модулем AudioDecoder для синтезу великомасштабної спектограми.

$$Y_{1:F,2:T+1} = AudioDec(R`) \tag{4.3}$$

Результат $Y_{1:F,2:T+1}$ порівняється з часовими змінами $S_{1:F,2:T+1}$.

Використовувані в моделі мережі повністю згорткові і не залежать відбудь-яких повторюваних одиниць. Замість RNN іноді використовуються переваги розширеної згортки, щоб враховувати довгу контекстуальную інформацію.

Рівняння на рисунку 4.6 є вмістом TextEnc.

$$\mathsf{TextEnc}(L) := (\mathsf{HC}^{2d \leftarrow 2d}_{1\star1})^2 \triangleleft (\mathsf{HC}^{2d \leftarrow 2d}_{3\star1})^2 \triangleleft (\mathsf{HC}^{2d \leftarrow 2d}_{3\star27} \triangleleft \mathsf{HC}^{2d \leftarrow 2d}_{3\star9} \triangleleft \mathsf{HC}^{2d \leftarrow 2d}_{3\star3} \triangleleft \mathsf{HC}^{2d \leftarrow 2d}_{3\star1})^2 \triangleleft \mathsf{C}^{2d \leftarrow 2d}_{1\star1} \triangleleft \mathsf{ReLU} \triangleleft \mathsf{C}^{2d \leftarrow e}_{1\star1} \triangleleft \mathsf{CharEmbed}^{e\text{-dim}}(L).$$

$$\mathsf{AudioEnc}(S) := (\mathsf{HC}^{d \leftarrow d}_{3\star3})^2 \triangleleft (\mathsf{HC}^{d \leftarrow d}_{3\star27} \triangleleft \mathsf{HC}^{d \leftarrow d}_{3\star9} \triangleleft \mathsf{HC}^{d \leftarrow d}_{3\star3} \triangleleft \mathsf{HC}^{d \leftarrow d}_{3\star1})^2 \triangleleft \mathsf{C}^{d \leftarrow d}_{1\star1} \triangleleft \mathsf{ReLU} \triangleleft \mathsf{C}^{d \leftarrow d}_{1\star1} \triangleleft \mathsf{ReLU} \triangleleft \mathsf{C}^{d \leftarrow F}_{1\star1}(S).$$

$$\mathsf{AudioDec}(R') := \sigma \triangleleft \mathsf{C}^{F \leftarrow d}_{1\star1} \triangleleft (\mathsf{ReLU} \triangleleft \mathsf{C}^{d \leftarrow d}_{1\star1})^3 \triangleleft (\mathsf{HC}^{d \leftarrow d}_{3\star1})^2 \triangleleft (\mathsf{HC}^{d \leftarrow d}_{3\star27} \triangleleft \mathsf{HC}^{d \leftarrow d}_{3\star9} \triangleleft \mathsf{HC}^{d \leftarrow d}_{3\star3} \triangleleft \mathsf{HC}^{d \leftarrow d}_{3\star1}) \triangleleft \mathsf{C}^{d \leftarrow 2d}_{1\star1}(R').$$

$$\mathsf{SSRN}(Y) := \sigma \triangleleft \mathsf{C}^{F' \leftarrow F'}_{1\star1} \triangleleft (\mathsf{ReLU} \triangleleft \mathsf{C}^{F' \leftarrow F'}_{1\star1})^2 \triangleleft \mathsf{C}^{F' \leftarrow 2c}_{1\star1} \triangleleft (\mathsf{HC}^{2c \leftarrow 2c}_{3\star1})^2 \triangleleft \mathsf{C}^{2c \leftarrow c}_{1\star1} \triangleleft (\mathsf{HC}^{c \leftarrow c}_{3\star3} \triangleleft \mathsf{HC}^{c \leftarrow c}_{3\star1} \triangleleft \mathsf{D}^{c \leftarrow c}_{2\star1})^2 \triangleleft (\mathsf{HC}^{c \leftarrow c}_{3\star3} \triangleleft \mathsf{HC}^{c \leftarrow c}_{3\star1}) \triangleleft \mathsf{C}^{c \leftarrow F}_{1\star1}(Y).$$

Рисунок 4.6 – Деталі компонента Text2Mel

Воно складається з вкладення символів і складної односпрямованої згортки. AudioEnc і AudioDec, показані на рисунку 4.5, складаються з 1D каузальних шарів згортки з активацією Highway. Ці згортки повинні бути причинними, оскільки висновок AudioDec звертається до введення AudioEnc на етапі синтезу .

Тепер розглянемо модель Spectrogram Super-resolution Network (SSRN). Тут синтезується повна спектрограма ¿$Z \vee \in R^{F' \times 4T}$, за отриманою великомасштабною спектрограмою спектра $Y \in RF \times T$, мережею спектрограм (SSRN). Частота дискретизації від F до F 'досить проста. Цього можна домогтися, збільшуючи

канали згортки 1D згортуючої мережі. Відтворення в часовому напрямку не виконується аналогічним чином, але, двічі застосовуючи шари деконволюції розміру кроку 2, в чотири рази збільшується довжина послідовності від T до 4T = T '. Нижне рівняння на малюнку 4.6 показує SSRN. Функція втрат така ж, як Text2Mel: сума бінарної розбіжність і відстань L1 між синтезованою спектрограмой SSRN (S) і істиною | Z |.

В цілому, "модуль уваги" досить дорогий для навчання. Тому, якщо є деякі попередні знання, то необхідно їх включити в модель, щоб полегшити процес навчання. На стадії синтезу матриця уваги, що іноді може неправильно зчитувати деякі символи. Типовими помилками, які спостерігаються найчастіше є:

–іноді пропускаються кілька символів;

–неодноразово читається одне і те ж слово двічі або більше;

Щоб зробити систему більш надійною необхідно евристично модифікувати матрицю A, як “майже діагональну”. Після цього спостерігається, що пристрій іноді починає пом'якшувати такі помилки [14].

4.4 Метод синтезу голосового мовлення Deep Voices

Deep Voice - це система перетворення тексту в мову, побудована виключно з глибоких нейронних мереж. Deep Voice закладає основу для справжнього синтезу емоційній промові. Система включає в себе п'ять основних блоків: модель сегментації для визначення меж фонем, модель перетворення графеми-в-фонеми, модель прогнозування тривалості фонеми, модель прогнозування основної частоти і модель синтезу звуку. Для моделі сегментації авторами методу був обраний новий спосіб обчислення меж фонеми за допомогою глибоких нейронних мереж, що використовують втрату тимчасової класифікації з'єднань (CTC). Для моделі синтезу звуку використовується варіант WaveNet, який вимагає менше параметрів і навчається швидше оригіналу. Використовуючи нейронну мережу для кожного компонента, система є більш простою і гнучкої, ніж традиційні текстові системи, де кожен компонент вимагає кропіткої розробки функцій і великої експертизи. Нарешті, заявлено, що вихідний сигнал даної системи може генеруватися швидше,

ніж в реальному часі, і характеризує оптимізовані ядра виведення WaveNet як на процесорах, так і на GPU, які досягають 400 кратного прискорення в порівнянні з існуючими реалізаціями. Загальну схему архітектури методу DeepVoice можна побачити на рисунку 3.6.

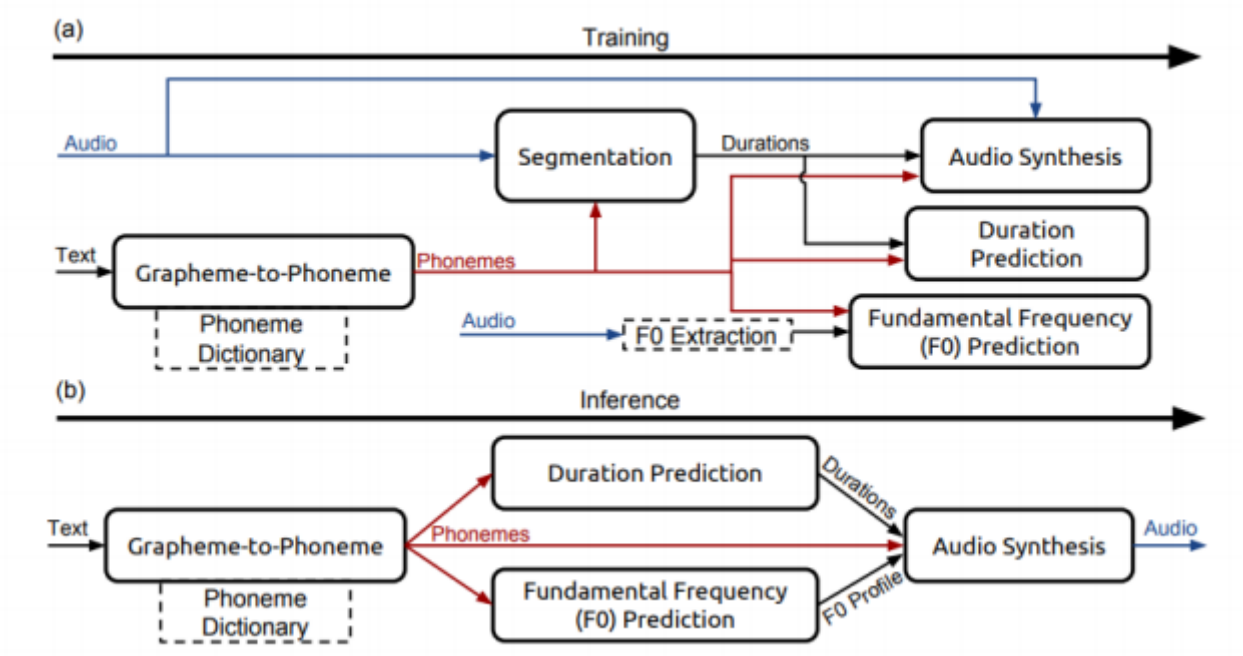

Рисунок 3.6 – Архітектура DeepVoice

Модель grapheme-to-phoneme. Вона перетворює з письмового тексту (англійські символи) в фонеми (кодується з використанням фонематичного алфавіту, такого як ARPABET).

Модель сегментації. Вона визначає межі фонем в голосові даних. З огляду на аудіофайли і транскрипцію аудіофайлу по фонемі, компонент сегментації ідентифікує, де в звуці кожна фонема починається і закінчується.

Модель тривалості фонеми. Вона передбачає тимчасову тривалість кожної фонеми в послідовності фонем (вимова). Основна частотна модель. Вона передбачає, озвучена чи фонема. Якщо це так, модель пророкує основну частоту (F0) по всій тривалості фонеми.

Модель синтезу звуку об'єднує виходи графеми-до-фонеми, тривалість фонеми і моделі прогнозування основної частоти і синтезує аудіо з високою частотою дискретизації, що відповідає бажаному тексту.

Під час виведення текст передається по моделі grapheme-tophoneme або словник фонеми для створення фонем. Далі, фонеми представлені в якості вхідних даних, для моделі тривалості фонем і моделі передбачення F0 для призначення тривалості кожної фонемі і формування контуру F0. Нарешті, фонеми, тривалості фонем і F0 використовуються в якості локальних функцій введення звуку для моделі синтезу звуку, яка генерує остаточне висловлювання. На відміну від інших моделей, модель сегментації не використовується під час виведення. Замість цього вона використовується для анотації навчальних голосових даних з межами фонем. Межі фонем на увазі тривалість, яку можна використовувати для навчання моделі тривалості фонем. Звук, анотований фонемами і тривалістю фонем, а також основною частотою, використовується для навчання моделі синтезу звуку.

Розглянемо кожен блок більш докладно. Модель grapheme-tophoneme заснована на архітектурі encoderdecoder, розробленої Yao & Zweig (2015). У моделі використовується багатошаровий двонаправлений кодер з не лінійністю стробірованний рекурентного блоку (GRU) і однаково глибокий односпрямований декодер GRU (Chung et al., 2014 року). Початковий стан кожного рівня декодера ініціалізуються кінцевим прихованим станом відповідного верхнього рівня кодера. Архітектура навчається за допомогою вчителя, а декодування виконується з використанням пошуку променю. У модулі використовується 3 двонапрямлених шару з 1024 одиницями кожен в кодере і 3 односпрямованих шару однакового розміру в декодере, і пошук променю з шириною 5 кандидатів. Під час навчання використовується виключення з ймовірністю 0,95 після кожного рекурентного шару. Для навчання використовується алгоритм оптимізації Адама з $\beta 1 = 0,9$, $\beta 2 = 0,999$, $\varepsilon = 10\text{-}8$, розмір пакета 64, швидкість навчання 10-3 і швидкість відпалу 0,85, яка застосовується кожні 1000 ітерацій (Kingma & Ba, 2014 року)

Модель сегментації навчається обчислювати вирівнювання між даним виступом і послідовністю цільових фонем. Це завдання аналогічне проблемі узгодження мови з текстовим виходом в розпізнаванні мови. У цій сфері функція втрати часу з'єднання (CTC) була показана для фокусування на вирівнюванні символів, щоб дізнатися відповідність між звуком і текстом (Graves et al., 2006).

Згорткова рекурентна архітектура нейронної мережі була адаптована з сучасної системи розпізнавання мови (Amodei et al., 2015) для виявлення меж фонеми.

Мережа, навчена CTC для генерації послідовностей фонем, буде давати короткі піки для кожної вихідної фонеми. Хоча цього достатньо, щоб грубо вирівняти фонеми з аудіо, цього недостатньо для визначення точних меж фонем. Щоб подолати це, необхідно навчати передбачати послідовності пар фонем, а не поодинокі фонеми. Потім мережа буде намагатися виводити пари фонем по таймаут, близьким до кордону між двома фонемами в парі. Щоб проілюструвати кодування міток, розглянемо рядок “Hello!”. Щоб перетворити її в послідовність парних міток фонем, перетворимо висловлювання в фонеми (використовуючи словник вимови, такий як CMUDict або модель grapheme-tophoneme) і прокладемо послідовність фонем з обох кінців з допомогою фонеми тиші, щоб отримати “тиша HH EH L OW тиша” нарешті, побудуємо послідовні пари фонем і отримаємо “(тиша, HH), (HH, EH), (EH, L), (L, OW), (OW, тиша)” .Вхідний аудіофрагмент реалізований шляхом обчислення 20 мел-частотних кепстральних коефіцієнтів (MFCC) з кроком в десять мілісекунд. у верхній частині вхідного шару є два шари згортки (2D згортки за часом і частоті), три двонапрямлених повторюваних шарів GRU і, нарешті, шар виведення softmax. Рівні згортки використовують ядра з одиничним кроком, висотою дев'ять (в частотних осередках) і шириною п'ять (за часом), а повторювані шари використовують 512 GRU комірок (для кожного напрямку). Виняток з ймовірністю 0,95 застосовується після останньої згортки і повторюваних шарів. Щоб обчислити частоту помилок фонем-пари (PPER), відбувається декодування з використанням пошуку променю. Щоб декодувати кордону фонем, виконується пошук променю з шириною 50 з обмеженням, що сусідні пари фонем перекриваються щонайменше однієї фонемой і відслідковуються позиції у висловленні кожної пари фонем. У даній моделі використовується єдина архітектура, щоб спільно прогнозувати тривалість фонеми і залежну від часу основну частоту.

Вхідними даними в модель є послідовність фонем з акцентами, причому кожна фонема і акцент кодуються як один гарячий вектор. Архітектура включає в себе два повністю пов'язаних шари з 256 одиницями кожен, за яким слідують два

односпрямованих повторюваних шарів з 128 осередками GRU кожен і, нарешті, повністю пов'язаний вихідний рівень. Виняток з ймовірністю 0,8 застосовується після початкових повністю підключених шарів і останнього повторюваного шару.

Останній рівень дає три оцінки для кожної вхідної фонеми: тривалість фонеми, ймовірність того, що фонема озвучена (тобто має основну частоту), і 20 залежних від часу значень F0, які вибірково відбираються по прогнозованій тривалості. Модель оптимізована за рахунок мінімізації спільної втрати, яка поєднує в собі похибка тривалості фонеми, помилку основної частоти, негативну логарифмічну ймовірність, ймовірності того, що фонема озвучена, і штрафний елемент, пропорційний абсолютній зміні F0 за часом для накладення гладкості.

Модель синтезу звуку - це варіант WaveNet. WaveNet складається з кондиціонуючою мережі, яка підвищує лінгвістичні 60 характеристики до бажаної частоти і мережі авторегресії, яка генерує розподіл ймовірності P (y) по дискретизованої зразкам аудіо y ∈ {0,1, ..., 255}. У моделі змінюється кількість шарів l, кількість залишкових каналів r (розмір прихованого стану кожного шару) і кількість пропущених каналів s (розмір, до якого виходять рівні до рівня вихідного рівня). WaveNet складається з мережі з підвищувальної дискретизацией і кондиціонуванням, наступних за l 2 × 1 згортальних шарів з r залишковими вихідними каналами і стробірованний нелінійними нелинейностями. Згортка розбивається на два матричних множення за один раз з W prev і Wcur. Ці шари пов'язані із залишковими сполуками. Приховане стан кожного шару конкатенуються з вектором r і проектується на s пропукая канали CW skip, за яким слідують два шари з 1 × 1 згорток (з вагами Wrelui Wout) [15].

4.5 Висновки до розділу 4

В даному розділі були розглянуті найбільш популярні методи синтезу мовлення, їхні архітектури, та особливості роботи нейронних мереж, на основі яких вони були побудовані,

5 ДАНІ ДЛЯ НАВЧАННЯ

Хороша база для реалізації синтезу – це основна запорука успіху. До підготовки нового голосу підходять дуже серйозно. Професійний диктор вимовляє заздалегідь підготовлені фрази протягом багатьох годин. Для кожного проголошення потрібно витримати всі паузи, говорити без ривків і пауз, відтворити правильний контур основного тону і все це в поєднанні з правильною інтонацією. Крім усього іншого, не всі голоси однаково приємно звучать.

В якості навчального набору даних ми вибрали LJSpeech dataset, який містить 13 100 аудіо записів по 2-10 сек. і файл з текстом, відповідаючий англійській мові, записаної на аудіо.

6 ОСНОВНІ ЗАСОБИ РОЗРОБКИ

6.1 Бібліотека TensorFlow

Основним програмним засобом глибокого навчання є TensorFlow. Це бібліотека з відкритим кодом штучного інтелекту, що використовує графіки потоків даних для побудови моделей. Це дозволяє розробникам створювати великомасштабні нейронні мережі з багатьма шарами. TensorFlow використовується в основному для: класифікації, сприйняття, розуміння, виявлення, прогнозування і створення.

Основні випадки використання TensorFlow.

Одним з найбільш відомих застосувань TensorFlow є програми на основі звуку. При належному подачі даних нейронні мережі здатні розуміти аудіосигнали.

Це можуть бути:

– Розпізнавання голосу - в основному використовується в IoT, Automotive, Security і UX / UI

– Голосовий пошук - в основному використовується в телекомунікаціях, виробниками телефонів

– Виявлення дефектів (шум двигуна) - в основному використовується в автомобільній та авіаційній галузях

Що стосується поширених випадків використання, всі ми знайомі з голосовим пошуком і голосовими асистентами з новими широкими поширеними смартфонами, такими як Siri компанії Apple, Google Now для Android і Microsoft Cortana для Windows Phone.

Розуміння мови є ще одним поширеним випадком використання для розпізнавання голосу. Програми мовлення в текст можна використовувати для визначення фрагментів звуку в більших аудіо файлах і транскрибування розмовного слова як тексту.

Звукові додатки також можуть використовуватися в CRM. Можливим є сценарій використання: алгоритми TensorFlow, що знаходяться в агентах

обслуговування клієнтів, і спрямовують клієнтів до потрібної інформації, і вони швидше, ніж агенти.

Текстові програми. Подальше популярне використання TensorFlow – це текстові програми, такі як сентиментальний аналіз (CRM, соціальні медіа), виявлення загроз (соціальні медіа, уряд) та виявлення шахрайства (страхування, фінанси)

Виявлення мови є одним з найбільш популярних варіантів використання текстових програм.

Ми всі знаємо Google Translate, який підтримує понад 100 мов, які перекладаються з одного на інший. Розроблені версії можуть бути використані для багатьох випадків, наприклад, перекладу жаргону юридичною у контракти на звичайну мову.

Підсумовування тексту. Google також виявив, що для більш коротких текстів, узагальнення можна вивчити за допомогою методики, яка називається послідовним навчанням. Це може бути використано для створення заголовків новинних статей.

Алгоритми TensorFlow Time Series використовуються для аналізу даних часових рядів з метою отримання значущої статистики. Вони дозволяють прогнозувати неспецифічні періоди часу, а також генерувати альтернативні версії часових рядів.

Використання, які впливатимуть один на одного і сприятимуть машинної технології навчання.

Найпоширенішим випадком використання часових рядів є Рекомендації. Ви, мабуть, чули про це використання від Amazon, Google, Facebook і Netflix, де вони аналізують діяльність клієнтів і порівнюють його з мільйонами інших користувачів, щоб визначити, що клієнт може купити або подивитися. Ці рекомендації стають ще більш розумними, наприклад, вони пропонують вам певні речі, як подарунки (не для себе) або телевізійні шоу, які можуть сподобатися членам вашої родини.

Інші способи використання алгоритмів TensorFlow Time Series в основному представляють інтерес для фінансів, бухгалтерського обліку, уряду, безпеки та IoT з виявленням ризиків, прогнозним аналізом та планування підприємств / ресурсів..

Нейронні мережі TensorFlow також працюють на відеоданих. Це в основному використовується в виявленні руху, виявлення потоків в реальному часі в іграх, безпеці, аеропортах і полях UX / UI. Нещодавно університети працюють над широкомасштабними наборами даних про класифікацію відео, таких як YouTube-8M, з метою прискорення досліджень щодо широкомасштабного розуміння відео, навчання представництв, шумного моделювання даних, передачі навчання та адаптації домену для відео.

Оскільки TensorFlow є бібліотекою з відкритим кодом, ми скоро побачимо ще багато інноваційних випадків [4].

6.2 Бібліотека Numpy

NumPy це open-source модуль для python, який надає загальні математичні і числові операції у вигляді пре-скомпільовані, швидких функцій. Вони об'єднуються в високорівневі пакети. Вони забезпечують функціонал, який можна порівняти з функціоналом MatLab. NumPy (Numeric Python) надає базові методи для маніпуляції з великими масивами і матрицями. SciPy (Scientific Python) розширює функціонал numpy величезною колекцією корисних алгоритмів, таких як мінімізація, перетворення Фур'є, регресія, і інші прикладні математичні техніки.

Головною особливість numpy є об’єктний масив. Масиви схожі з списками в Python, виключаючи той факт, що елементи повинні мати одинаковий тип даних, як float і int. З массивами можна проводити числові операції з великим об’ємом інформації в рази швидше і що головне більш ефективніше чим зі списками. Масиви можуть бути і багатовимірними

6.3 Бібліотека Matplotlib

Matplotlib – бібліотека на мові програмування Python для візуалізації даних двовимірною 2D графікою (3D графіка також підтримується). Отримувані зображення можуть бути використані як ілюстрації в публікаціях.

Зображення, які генеруються в різних форматах, можуть бути використані в інтерактивній графіці, наукових публікаціях, графічному інтерфейсі користувача, веб-додатках, де потрібно будувати діаграми.

Бібліотека Matplotlib побудована на принципах ООП, але має процедурний інтерфейс pylab, який надає аналоги команд MATLAB.

Пакет підтримує багато видів графіків і діаграм:

– графіки;
– діаграми розсіювання;
– стовпчасті діаграми і гістограми;
– секторні діаграми;
– діаграми “стовбур-листя”;
– контурні графіки;
– поля градієнтів;
– спектральні діаграми.

6.4 Висновки до розділу 6

В даному розділі були розглянуті необхідні та найбільш використовуванні засоби розробки які використовуються при розробці нейронних мереж.

Також при установці всіх потрібних пакетів, для підтримання роботи систему не однократно зустрічалися проблеми, тому найкращою рекомендацією буде, проводження всіх робіт на оперативній системі Linux, це зекономить час підготовки, оскільки якщо робити це на оперативній системі Windows, я велика ймовірність що деякі пакети не будуть, працювати.

Також варто взяти до уваги, що при не коректній версії NumPy, бібліотека TensorFlow, не буде працювати, тому краще спочатку, встановлювати саме TensorFlow, вона автоматично встановить потрібну версію для її підтримки.

Не рекомендується використовувати нову версію Python 3.7, тому що не всі бібліотеки, фреймворки, налаштовані до нових змін в файловій системі, яка використовуються в Python 3.7 , що призводить до незрозумілих помилок, які важко відслідкувати.

7 ДОСВІД НАВЧАННЯ

Першим розглянутим методом автоматичного синтезу мови в даному дослідженні був WaveNet. Підводячи підсумки можна сказати, що це дуже дорогий метод, так як необхідно зробити велику кількість обчислень, тобто потрібно потужне обладнання, 67 яке є недешевим. Типова WaveNet для високоякісної мови використовує 40 шарів згортки, поряд з іншими зв’язками між ними. І так як один зразок генерується за раз, тому для генерації 1 секунди звуку 16 кГц потрібна обробка 16000 вибірок. Після обчислень виходить, що для створення 1 секунди аудіо потрібно близько 4 хвилин. Це не прийнятна система синтезу мови для онлайн використання, і це відповідно знижує спектр її застосування, адже висновок в режимі реального часу є вимогою для системи якості TTS.

Але WaveNet довів, що якість безпосередньо генеруючих звукових зразків значно вище, ніж використання вокодера з ручними конструктивними функціями. Адже, не потрібно оцінювати фазу, інтонацію, стрес та інші аспекти мови, необхідно просто знайти кращу архітектуру, яка буде генерувати відповідні зразки. Нижче наведена узагальнена порівняльна таблиця 3.1 на підставі дослідження MOS.

Таблиця 7.1 – Порівння синтизованого звуку WaveNet з стандартними синтезаторами і людським мовленням

Метод	Голос 1	Голос 2	Голос 3	Голос 4
Стандартний (без використання нейронних мереж)	3,18	3,44	3,65	3,33
WaveNet	3,99	4,22	4,28	3,98
Людське мовлення	4,52	4,44	4.59	4,11

MOS є стандартною мірою для оцінювання суб'єктивних тестів якості звуку і виходить в сліпих тестах з людьми (від більш ніж 500 оцінок на 100 тестових

пропозицій). При виставленні оцінки счіатется, що 0 - взагалі не була синтезована, а 5 - мова синтезована ідеально. Результати були статистично оброблені.

Як видно з таблиці 3.1, що вийшла в результаті система Text-toSpeech забезпечила якісний голос. Також можна відзначити, що вихідні сигнали виходять дуже чистими (без шуму), без гудіння, без приглушеної мови [18].

Результат дослідження методу DCTTS перевершив всі очікування за якістю синтезованого мовлення . Система розставляє наголоси, мова виходить розбірливою, а голос зрозумілою. Також цей великим плюсом цього методу є те, що в порівнянні з WaveNet він вимагає набагато менше даних для навчання, мову середньої якість можна отримати використовую всього 5 годин навчальних даних. Також було створено порівняння згенерованої мови з методом Tacotron, і DCTTS для кращої оцінки якості в дослідженні були використані різні кількості ітерацій для модулів Text2Mel і SSRN, і різний час навчання мережі. Отримані результати відображені в таблиці 7.2.

Таблиця 7.2 – Порівняння якості синтезованої мови Tacotron и DCTTS

Метод	Кількість ітерацій (Text2Mel,SSRN)	Час навчання	Середня оцінка MOS
Tacotron	887K	12днів	2,07 ±0,62
DCTTS	20K / 40K	~ 2 години	1,74 ± 0,52
DCTTS	90K / 150K	~ 7 години	2,63 ± 0,75
DCTTS	200K / 340K	~ 15 години	2,71 ± 0,66
DCTTS	540K / 900K	~ 40 години	2,61 ± 0,62

Рекурентна нейронна мережа (RNN) була стандартною методикою для моделювання послідовних даних останнім часом цей метод використовується в передових нейронних методах TTS. Однак для підготовки компонента RNN часто потрібно дуже потужний комп'ютер або дуже тривалий час, як правило, кілька тижнів. Недавні дослідження, показали, що синтез послідовності на основі CNN може бути набагато швидше, ніж методи на основі RNN, через високу паралельність. Це було доведено даним дослідженням, адже було виявлено, що

Deep Convolutional TTS буде навчений до прийнятної якості мови за ~ 15 годин, використовуючи звичайний ігровий ПК, оснащений двома графічними процесорами [17]. Нижче наведена порівняльна таблиця методів

Таблиця 7.3 – Порівняння якості методів синтезу мовлення

Метод	WaveNet	Tacotron	DeepVoice	DCTTS
Архітектура	WaveNet	Tacotron		Tacotron з модифікуванням
Нейронна мережа	CNN	RNN	RNN	CNN
Залежність	Кількість згорток	mel та лінійних спектрограм	mel та лінійних спектрограм	Кількості згорток, спектральних представлень аудіо
Кількість шарів	40+	-	-	30+
Час на створення 1 секунди аудио	~240 секунд	~30 секунд	~0,5 секунд	~12 секунд
Час необхідний на тренування	1 тиждень	2тиждні	2тижні	4-5 днів
Кількість вхідних даних	24-34 години	20 годин	10днів	24години

При пошуку реалізації існуючої нейронної мережі, основним варіантом було вибрано саме DCTTS, оскільки архітектура була схожа з Tacotron 2, проте швидкість навчання даної системи, по проаналізованим даним, складала менший відрізок часу.

Використовуючи дану систему була спроба навчання нейронної мережі, при спробі по 20 тисяч ітерацій, приблизно дві години, тренування, було виявлено, що дана кількість ітерацій є малою, і система генерувала тільки однорідний писк, або тишу .

При збільшені ітерації до 50 тисяч, приблизно 15годин для Text2Mel,а для SSRN 5 годин, голос, став зрозумілий, проте все ж таки залишилося відчуття, того, що голос, ще не є повністю природним (рисунок 7.1).

Подальші експерименти не відбувалися у зв'язку з високою затратністю даних способів, як календарного часу, так і машинного. При чому при аналізуванні результатів, інших дослідників, було виявлено, щоб натренувати систему до бажаного результату, потрібно близько 2 тижнів. Проте це ще не гарантований результат успіху, потрібні ще правильно підібрані гіпер-параметри .

Тому було вирішено взяти уже натреновану модель, яка була натренована на 200 тисяч ітерацій, для створення веб-застосунку

```
# training scheme
lr = 0.001 # Initial learning rate.
logdir = "logdir/LJ01"
sampledir = 'samples'
B = 32 # batch size
num_iterations = 50000
```

Рисунок 7.1 – Параметри фінального тренування системи

Проте був створений веб-застосунок, для можливості використання, даної натренованої моделі. Який записував, текст набраний користувачем, в полі призначеному для вводу тексту, в файл. Після підтвердження користувачем введеної фрази, на сервері створювався, файл, та запускалася, натренована нейронна мережа, яка генерувала, відповідну тексту, аудіо фразу, зразу же після генерації файлу, на стороні сервера, файл формату .wav, буде надісланий до користувача.

Проте в даного сервісу є проблема, яка вирішується збільшенням потужності сервера, оскільки час на генерацію одного речення, яке становить не більше 180

символів, може зайняти, близько хвилини, то й більше, що є недопустимо, в реаліях нашого часу.

Хоча якість звуку ще далека від досконалості, можливість, вдосконалювалася шляхом точного налаштування деяких гіпер-параметрів і застосуванням деяких методів, розроблених в співтоваристві дослідників в цій області. Тому я вважаю, що цей зручний метод сприяє подальшому розвитку програм, заснованих на синтезі мови[3].

В даному розділі було проведена аналітика стосовно існуючих і провідних методів систем TTS. Були розглянуті всі їхні переваги та недоліки, розглянуті всі особливості роботи, на основі цього було вибраний метод DCTTS, який займає найменше часу для тренування, та середній для генерації 1секунди тексту

На основі цього був створений веб-застосунок, який можна використовувати для генерації аудіо файлів, по заданому тексту в реальному часі, проте для роботи потрібно збільшити обчислювальну швидкість сервера, для оптимальної швидкості генерації тексту, для користувача, який захоче користуватися даним сервісом.

Тобто як висновок, системи TTS, в реальному часі це наше найближче майбутнє , особливо якщо збільшиться обчислювальна можливість комп’ютерів.

ВИСНОВКИ

В ході виконання даної роботи було проведено дослідження у сфері синтезу мовлення , в ході роботи було проаналізовано методи синтезу голосового мовлення , були знайдені рішення для даної проблеми , ним виявилися саме нейронні мережі, які зарекомендували себе в багатьох сферах. Проаналізувавши їх архітектури, переваги та недоліки, був досягнутий висновок, що для генерації мовлення, потрібні велика кількість обчислювальних ресурсів , проте генерований текст схожий на то, я б записала людина.

За допомогою DCTTS, вдалося створити власну систему перетворення тексту в мовлення, на англійській мові, оскільки на українській мові, не було підготовлених дата-сетів у вільному доступі.

Також був створений веб-застосунок, який дає можливість генерації з тексту аудіо файл. Веб-застосунок був створений на основі архітектури “клієнт-сервер”,з не вирішених проблем залишилася швидкість генерації аудіо файлу.Проте збільшивши потужність сервера, який генерує дані, можливо отримати оптимальний час, для користувача.

Кінцевими користувачами розробленої системи є будь-хто ,бажаючий перетворити інформацію з одного формату в інший.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кривонос Ю.Г., Крак Ю.В., Шатковский Н.Н. Анализ структуры задачи создания систем озвучивания текстовой информации // Компьютерная математика. – 2005. – № 3.

2. TTS System [Електронний ресурс] – Режим доступу до ресурсу: http://www.research.att.com/projects/tts/.

3. Siri does more than ever. Even before you ask. [Електронний ресурс] – Режим доступу до ресурсу: https://www.apple.com/siri/.

Watson Text to Speech [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/cloud/watson-text-to-speech?mhq=TTS&mhsrc=ibmsearch_a.

4. Dutoit T. and Leich H.. MBR-PSOLA: Text-to-speech synthesis based on an MBE re-synthesis of the segments database. Speech Communication, 1993.

5. Cox R.V., Kamm C.A., Rabiner L.R., Schroeter J., Wilponl J.G. Speech and language processing for next-millennium communications services // Proceeding of the IEEE. – Vol. 88, № 8, august 2000.

6. Слушать сайты в любое время и в любом месте [Електронний ресурс]. – 2008. – Режим доступу до ресурсу: https://habr.com/ru/post/23034/

7. Нейросетевой синтез речи своими руками [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://habr.com/ru/company/speechpro/blog/358816/?fbclid=IwAR334TsZwgIXgOMR-EHWlm0Yuz6tF105k0PSmcTmX5QW5b-bkm2xH9fumE.

8. Людовик Т. В. ИСПОЛЬЗОВАНИЕ РЕЧЕВЫХ БАЗ ДАННЫХ БОЛЬШОГО ОБЪЕМА ПРИ СИНТЕЗЕ РЕЧИ В СИСТЕМАХ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА [Електронний ресурс] / Т. В. Людовик, Н. Н. Сажок. – 2015. – Режим доступу до ресурсу: http://masters.donntu.org/2015/fknt/bakalenko/library/ludovick.pdf.

9. Katsuya U. EFFICIENTLY TRAINABLE TEXT-TO-SPEECH SYSTEM BASED ON DEEP CONVOLUTIONAL NETWORKS WITH GUIDED ATTENTION

[Електронний ресурс] / U. Katsuya, Hideyuki Tachibana. – 2017. – Режим доступу до ресурсу: https://arxiv.org/pdf/1710.08969.pdf.

10. Schwarz D. Current Research In Concatenative Sound Synthesis [Електронний ресурс] / Diemo Schwarz // Proceedings of the International Computer Music Conference. – 2005. – Режим доступу до ресурсу: http://recherche.ircam.fr/equipes/analyse-synthese/schwarz/publications/icmc2005/Schwarz_ICMC2005_Current-Research.pdf.

11. SYNTHESIZING EMOTIONS IN SPEECH: IS IT TIME TO GET EXCITED? [Електронний ресурс] – Режим доступу до ресурсу: https://pdfs.semanticscholar.org/6dd5/7a3aa2245f8961c4fa354a81d1f56a340e06.pdf.

12. Моделі нейронних мереж. – Режим доступу: https://studme.com.ua/1246122010028/neural/models.htm

13. Моделі нейронних мереж. – Режим доступу: http://techn.sstu.ru/kafedri/подразделения/1/MetMat/Terin/neiro/neiro.htm

14. Gradient-Based Learning Applied to Document Recognition [Електронний ресурс] – Режим доступу до ресурсу: http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf

15. Deep Convolutional Inverse Graphics Network [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/pdf/1503.03167v4.pdf.

16. Finding Structure in Time [Електронний ресурс] – Режим доступу до ресурсу: https://crl.ucsd.edu/~elman/Papers/fsit.pdf.

17. Введение в архитектуры нейронных сетей [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://habr.com/ru/company/oleg-bunin/blog/340184/.

18. Что такое свёрточная нейронная сеть [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/309508/.

ДОДАТОК А

Нейронні мережі для синтезу мовлення

Специфікація

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТР5278_19Б

Аркушів 2

Київ 2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТР5278_19Б	Записка.docx	Пояснювальна записка до дипломної роботи
Компоненти		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТР5278_19Б 12-1	server.js	Основний компонент
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТР5278_19Б 13-1	model_gs_718k.data -00000-of-00001	Натренована модель

ДОДАТОК Б

Нейронні мережі для синтезу мовлення

Текст програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТР5278_19Б

Аркушів 4

Київ 2019

Текст реалізації роута сервера для генерації тексту

```
/ API
const express = require("express");
const fs = require("fs");
const router = express.Router();
const path = require("path");
const child_process = require("child_process");

const pathToFile = "../TTS/harvard_sentences.txt";
const pathToAudioFile = "/home/alex/diploma/TTS/samples/1.wav";
const template = data => File with samples \n1. ${data};

router.post("/data", async (req, responce) => {
  try {
    const { payload } = req.body;
    if (!payload) return res.status(400).json("No payload data on req");

    await fs.writeFile(pathToFile, template(payload), "utf-8", err => {
      if (err) res.send(err);
    });

    await new Promise((res, rej) => {
      let workerProcess = child_process.spawn(
        "cd",
        ["/home/alex/diploma/TTS && python synthesize.py"],
        { shell: true }
      );

      workerProcess.stdout.on("data", data => {
        console.log("data", data);
        console.log("Start data .....");
        console.log(data.toString("utf-8"));
      });
      workerProcess.stdout.on("message", data => {
        console.log("data", data);
        console.log("message data .....");
        console.log(data.toString("utf-8"));
      });
      workerProcess.stdout.on("close", data => {
        console.log(data);
        console.log("Finished fetch data");
        console.log("stdout-Close");
        fs.readFile(pathToAudioFile, function(error, data) {
          console.log("kekos");
          if (error) {
            responce.status(404).send("Resourse not found!");
          } else {
            responce.send(data);
          }
        });
      });
      workerProcess.stdout.on("error", data => {
        console.log(data);
        console.log("ERROR data");
        console.log(data.toString("utf-8"));
        rej(data);
      });
      workerProcess.stderr.on("error", data => {
        console.log(data.toString("error"));
      });
      workerProcess.on("error", data => {
        console.log(data);
      });
    });
  } catch (error) {
    console.error(error);
  }
});

module.exports = router;
```

Текст програми генерації аудіо файлу з тексту

```
# -*- CODING: UTF-8 -*-
# /USR/BIN/PYTHON2
'''
BY KYUBYONG PARK. KBPARK.LINGUIST@GMAIL.COM.
HTTPS://WWW.GITHUB.COM/KYUBYONG/DC_TTS
'''

FROM __FUTURE__ IMPORT PRINT_FUNCTION

IMPORT OS

FROM HYPERPARAMS IMPORT HYPERPARAMS AS HP

IMPORT TENSORFLOW AS TF
IMPORT NUMPY AS NP
FROM TRAIN IMPORT GRAPH
FROM UTILS IMPORT *
FROM DATA_LOAD IMPORT LOAD_DATA
FROM SCIPY.IO.WAVFILE IMPORT WRITE
FROM TQDM IMPORT TQDM

DEF SYNTHESIZE():
    # LOAD DATA
    L = LOAD_DATA("SYNTHESIZE")

    # LOAD GRAPH
    G = GRAPH(MODE="SYNTHESIZE"); PRINT("GRAPH LOADED")

    WITH TF.SESSION() AS SESS:
        SESS.RUN(TF.GLOBAL_VARIABLES_INITIALIZER())

        # RESTORE PARAMETERS
        VAR_LIST = TF.GET_COLLECTION(TF.GRAPHKEYS.TRAINABLE_VARIABLES, 'TEXT2MEL')
        SAVER1 = TF.TRAIN.SAVER(VAR_LIST=VAR_LIST)
        SAVER1.RESTORE(SESS, TF.TRAIN.LATEST_CHECKPOINT(HP.LOGDIR + "-1"))
        PRINT("TEXT2MEL RESTORED!")

        VAR_LIST = TF.GET_COLLECTION(TF.GRAPHKEYS.TRAINABLE_VARIABLES, 'SSRN') + \
                   TF.GET_COLLECTION(TF.GRAPHKEYS.GLOBAL_VARIABLES, 'GS')
        SAVER2 = TF.TRAIN.SAVER(VAR_LIST=VAR_LIST)
        SAVER2.RESTORE(SESS, TF.TRAIN.LATEST_CHECKPOINT(HP.LOGDIR + "-2"))
        PRINT("SSRN RESTORED!")

        # FEED FORWARD
        ## MEL
        Y = NP.ZEROS((LEN(L), HP.MAX_T, HP.N_MELS), NP.FLOAT32)
        PREV_MAX_ATTENTIONS = NP.ZEROS((LEN(L),), NP.INT32)
        FOR J IN TQDM(RANGE(HP.MAX_T)):
            _GS, _Y, _MAX_ATTENTIONS, _ALIGNMENTS = \
                SESS.RUN([G.GLOBAL_STEP, G.Y, G.MAX_ATTENTIONS, G.ALIGNMENTS],
                         {G.L: L,
                          G.MELS: Y,
                          G.PREV_MAX_ATTENTIONS: PREV_MAX_ATTENTIONS})
```

```
            Y[:, J, :] = _Y[:, J, :]
            PREV_MAX_ATTENTIONS = _MAX_ATTENTIONS[:, J]

        # GET MAGNITUDE
        Z = SESS.RUN(G.Z, {G.Y: Y})

        # GENERATE WAV FILES
        IF NOT OS.PATH.EXISTS(HP.SAMPLEDIR): OS.MAKEDIRS(HP.SAMPLEDIR)
        FOR I, MAG IN ENUMERATE(Z):
            PRINT("WORKING ON FILE", I+1)
            WAV = SPECTROGRAM2WAV(MAG)
            WRITE(HP.SAMPLEDIR + "/{}.WAV".FORMAT(I+1), HP.SR, WAV)

IF __NAME__ == '__MAIN__':
    SYNTHESIZE()
    PRINT("DONE")
```

ДОДАТОК В

Нейронні мережі для синтезу мовлення

Опис програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТР5278_19Б

Аркушів 9

Київ 2019

АНОТАЦІЯ

Розділ містить опис частини, яка слугує для генерації мовлення по заданому тексту. Заданий застосунок використовує натреновану модель нейронної мережі, для генерації аудіо файлу . Та через http протокол надсилає до користувача.

Модуль написано мовою програмування JavaScript, з використанням Node.js та бібліотеки React.js.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

У цьому додатку міститься опис застосунку котрий вирішує проблему перетворення тексту в аудіо формат. У додатку Б міститься програмний код основних модулів розробленого застосунку.

Розроблена система може використовуватися будь-яким користувачем. Використання даного застосунку вимагає від користувача ПК, браузер та доступ до інтернету

Під час розробки системи використовувалася мова програмування JavaScript, середовище розробки VsCode, а також потужності платформи Node.js та бібліотеки React.js.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Розроблений застосунок використовується у випадках необхідності перетворення тексту мові в 32-бітний wav аудіо файл. Текст повинен вводитися користувачем лише на англійський мові, а також має обмеження в кількості 180 символів, для генерації аудіо в єдиний файл. Користувач може як прослухати так і завантажити на свій комп’ютер wav аудіо файл.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

При запуску системи генерування мовлення користувач бачить початкове вікно програми, в якому є поле для вводу тексту, дане поле має певні обмеження. Після того як користувач ввів текст, потрібно натиснути на кнопку, для записку процесу генерації тексту, після певного очікування. Користувач побачить перед собою аудіо файл, який він може прослухати або завантажити.

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Для забезпечення повноцінної роботи та досягнення високої ефективності роботи створеного додатку було обрано Visual Studio Code,яка показала себе надійним та гнучким редактором коду розробки програм.

Розроблений додаток працює в будь-якому браузері якій підтримує 32-бітні wav файли.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Розроблений застосунок потребує встановленої node.js та npm. Для запуску додатку потрібно перейти в папку і з кодом і в терміналі прописати команду npm run start , після цього на localhost:3000 можна використовувати даний застосунок

ВХІДНІ І ВИХІДНІ ДАНІ

Вхідними даними для розроблених додатків є текст на англійський мові який зчитується з textBox.

Вихідними даними є 32-бітний wav файл генерованого мовлення

ДОДАТОК Г

Нейронні мережі для синтезу мовлення

Апробації

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТР5278_19Б

Аркушів 3

Київ 2019

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

СУЧАСНІ ПРОБЛЕМИ НАУКОВОГО ЗАБЕЗПЕЧЕННЯ ЕНЕРГЕТИКИ

Матеріали XVII Міжнародної
науково-практичної конференції
молодих вчених та студентів
м. Київ, 23-26 квітня 2019 року,

ТОМ 2

Київ- 2019

УДК 004.52

Студент 4 курсу, гр. ТР-52 Вишняк О.М.
Доц., к.т.н. Стативка Ю.І.

НЕЙРОННІ МЕРЕЖІ ДЛЯ СИНТЕЗУ МОВЛЕННЯ

Синтез мови на сьогоднішній день застосовується в багатьох областях. Це і голосові асистенти, і IVR-системи (системи голосових меню), і розумні будинки, і т.і. Саме по собі завдання, на мою думку, досить зрозуміле: написаний текст, повинен вимовлятися так, як це б зробила людина.

Система голосового синтезування (також відома як функція перетворення «текст-мова», text-to-speech, TTS) зазвичай будується на базі одного з двох основних методів - конкатенативного та параметричного.

Конкатенативний (або компілятивний) метод передбачає побудову фраз шляхом збору окремих звукових фрагментів слів і частин, заздалегідь записаних із залученням актора озвучування. Основним недоліком такого методу є необхідність постійної заміни звукової бібліотеки щоразу, коли відбуваються якісь поновлення або вносяться зміни.

Інший метод носить назву параметричного TTS, і його особливістю є використання стилів, за допомогою яких комп'ютер генерує потрібну фразу. Мінус методу в тому, що найчастіше результат проявляється у вигляді нереалістичного або так званого роботизованого звучання.

Декілька років тому, в область синтезу мови, як і в багато інших областей, прийшло машинне навчання. З'ясувалося, що цілий ряд компонентів всієї системи можна замінити на нейронні мережі, що дозволяє не просто наблизитися за якістю до існуючих алгоритмів, а навіть значно їх перевершити. Розглянемо один із сучасних алгоритмів,який базується на нейронних мережах.

Робота алгоритму WaveNet полягає в поточковій генерації звукової хвилі за допомогою спеціальної нейромережі. Нейронна мережа, навчена на великій множині записів голосу диктора, не використовує окремі фрагменти та не зшиває їх, а генерує кожну точку звукового сигналу окремо. При створенні програми розробники використовували нейромережу типу FCN (Fully convolutional network) - нейронну мережу з повним згортанням. Кожен шар в цій мережі має свій фактор розширення, завдяки якому її рецептивне поле, тобто частина інформації, яку обробляють нейрони, зростає експоненційно. По суті, це дозволяє програмі охоплювати відразу велику кількість часових кроків. У нейромережі також передбачений зворотний зв'язок, тому кожен наступний звук машинної мови генерується на основі безлічі попередніх.

Перелік посилань:

1. WaveNet launches in the Google Assistant [Електронний ресурс] : [Вебсайт].– Режим доступу: https://deepmind.com/blog/wavenet-launches-google-assistant/ (дата звернення 12.03.2019).
2. Нейросетевой синтез речи своими руками [Електронний ресурс] : [Вебсайт].– Режим доступу: https://habr.com/ru/company/speechpro/blog/358816/ (дата звернення 12.03.2019).